1991-03

Enhanced productivity tools: an analysis of their procurement, implementation and operations

Lindenbaum, Eric James

Monterey, California. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

ENHANCED PRODUCTIVITY TOOLS : AN EVALUATION OF THEIR PROCUREMENT, IMPLEMENTATION AND OPERATIONS

by

Eric James Lindenbaum

March, 1991

Thesis Advisor: LCDR Robert Knight

Approved for public release; distribution is unlimited

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School 6b. OFFICE SYMBOL 37	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c. ADDRESS (*City, State, and ZIP Code*) Monterey, CA 93943-5000	7b. ADDRESS (*City, State, and ZIP Code*) Monterey, CA 93943-5000
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL *(If applicable)*	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (*City, State, and ZIP Code*)	10. SOURCE OF FUNDING NUMBERS Program Element No. / Project No. / Task No. / Work Unit Accession Number

11. TITLE (*Include Security Classification*)
ENHANCED PRODUCTIVITY TOOLS : AN ANALYSIS OF THEIR PROCUREMENT, IMPLEMENTATION AND OPERATIONS

12. PERSONAL AUTHOR(S) LINDENBAUM, ERIC JAMES

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (*year, month, day*)	15. PAGE COUNT
Master's Thesis	From To	MARCH, 1991	107

16. SUPPLEMENTARY NOTATION
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17. COSATI CODES			18. SUBJECT TERMS (*continue on reverse if necessary and identify by block number*)
FIELD	GROUP	SUBGROUP	Productivity; Software development; Database application generators.

19. ABSTRACT (*continue on reverse if necessary and identify by block number*)

Reductions in available Information Resources (IR) dollars in the budget places increased emphasis on the productivity of both system developers and users. New technologies have been proposed to improve these productivities. Three software development tools in particular have been proposed to the Navy. One was developed using Naval assets. Another is available Off-The-Shelf and the third was developed by both government assets and by a contractor. This paper will evaluate and compare the three products and their associated development techniques.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION	
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS REPORT ☐ DTIC USERS	Unclassified	
22a. NAME OF RESPONSIBLE INDIVIDUAL LCDR Robert Knight	22b. TELEPHONE (*Include Area code*) (408) 646-2771	22c. OFFICE SYMBOL AS/Kt

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted
All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

Approved for public release; distribution is unlimited.

ENHANCED PRODUCTIVITY TOOLS : AN ANALYSIS OF THEIR PROCUREMENT, IMPLEMENTATION AND OPERATIONS

by

Eric James Lindenbaum
Lieutenant , United States Navy
B.S., United States Naval Academy, 1981

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN INFORMATION SYSTEMS

from the

NAVAL POSTGRADUATE SCHOOL
March, 1991

ABSTRACT

Reductions in available Information Resources (IR) dollars in the budget places increased emphasis on the productivity of both system developers and users. Three software development tools in particular have been proposed to the Navy. One was developed using Naval assets. Another is available Off-The-Shelf and the third was developed using both government assets and by a contractor. This paper will evaluate and compare the three products and their associated development techniques.

TABLE OF CONTENTS

I. INTRODUCTION

A. PROBLEM STATEMENT

Reductions in available Information Resources (IR) dollars in the budget places increased emphasis on the productivity of both system developers and users. New technologies and techniques have been proposed to improve these productivities. In particular, fourth generation language (4GL) tools and developmental techniques designed to exploit them, all claim productivity enhancements.

Two Naval Regional Automatic Data Centers (NARDACs) have independently implemented different means to increase programmer and end user productivity. Both activities use a fourth generation language tool however, one is produced with in-house assets and the other is purchased off-the-shelf. The tools are the commercially available Paradox series and the NARDAC San Diego developed DB3GEN. A third alternative, which has been proposed by their headquarters, Naval Computers and Telecommunications Command (NCTC). While AdaSAGE does not use a fourth generation non-procedural language, it does encompass the same fourth generation development techniques as the others. This paper will evaluate and compare the three products and their associated development techniques.

The United States Navy desires to standardize its tools in an effort to obtain economies of scale in training and software procurement costs. The 4GL tool the Navy selects will influence the future of both the IR community and systems they create. The decision should be made only after a careful examination of many factors such as the tool's functionality, productivity and life cycle costs.

B. BACKGROUND

1. The Software Crisis

The world is experiencing a software crisis. The crisis is the inability of software technology to keep pace with hardware technology (Conte, 1986, p.1). The reasons for this crisis are many and complex:

- The growth in demand for higher quality, critical software in information-based and embedded code systems (Conte, 1896, p. 2)
- The productivity of programmers has not kept pace with the demand for completed applications (Martin, 1985, p. 2)
- While the decrease in the cost of hardware has made computer systems more available for different applications, the cost of the software to run these applications has been increasing (Boehm, 1987, p.44) Since 1973, there has been a 12 percent per year growth rate in the cost of developing software (Martin, 1985, p. 5)
- The number of computers and applications is growing. One estimate places the need for programming in 1995 at 372 times the 1985 level. Given the 1985 level of 400,00 programmers, 148 million programmers would be needed if their productivity remained constant. (Martin, 1985, p. 1) See figure 1.

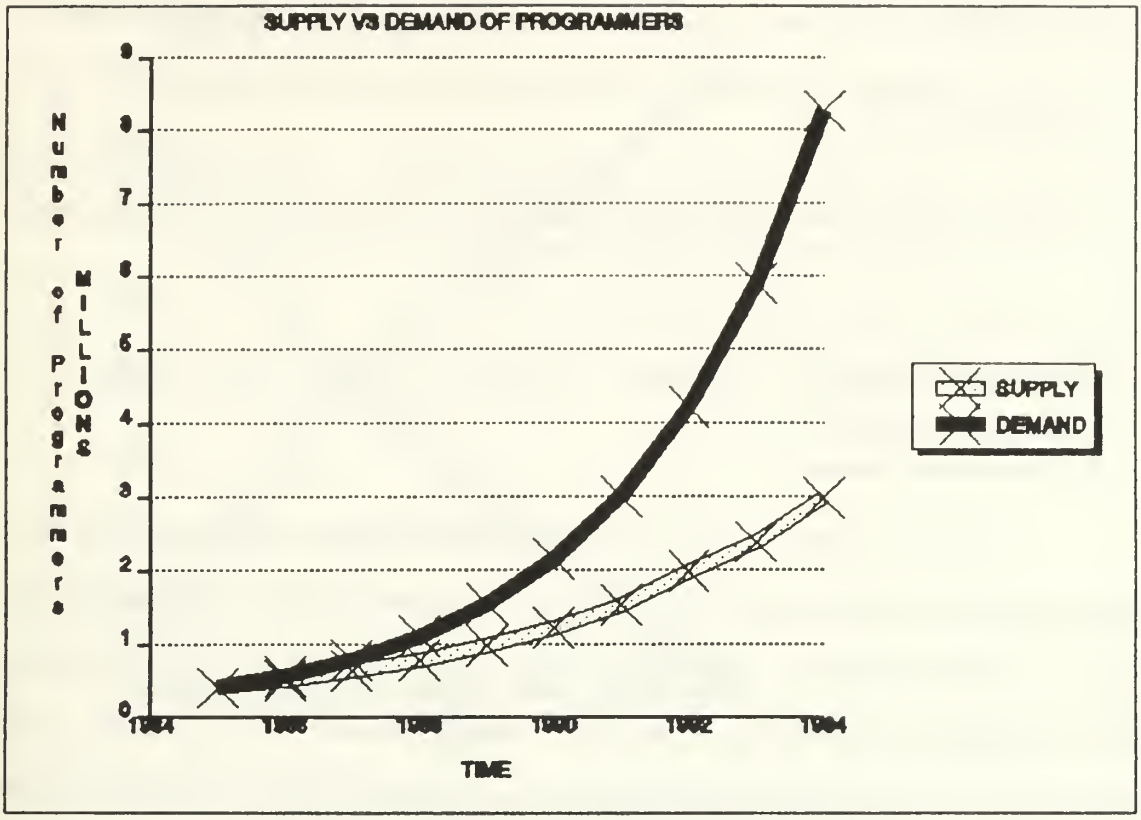

Figure 1 - Supply and Demand of Programmers

New technologies and methodologies are being marketed that claim to solve the productivity problem. Computer aided software engineering (CASE), 4GLs, application generators, and rapid prototyping are examples of types of products or methodologies designed to address the crisis. Selecting between these presents the information manager a decision which must be addressed in a structured manner. Following the structured analysis steps of Edward Yourdon, the situation facing the United States Navy is:

- The problem, a lack of a proper productivity level, has been identified

- The alternatives presented are the possible solutions
- Next, formulate the criteria for the evaluation of the alternatives
- After evaluating the alternatives, select the most advantageous solution (GSA Guide for Requirements Analysis and Analysis of Alternatives, p. 3-7, 1990)
- Implement and test the solution
- Maintain the new system. (Yourdon, pp. 42-64, 1988)

In 1960, the ratio of expenditures for hardware versus software was 80 percent hardware cost and 20 percent software cost. These hardware systems were all mainframe computers and their applications were mostly transactional processing. The introduction of the microcomputer and the growth of the number of software applications, especially into embedded systems, started changing the relationship. By 1980, the ratio had reversed to 80 percent now spent on software and only 20 percent on hardware. In the 1990's the trend will continue with over 90 percent being eventually spent software alone. See figure 2. Software, not hardware now drives the overall life cycle costs of a computer system. (Fairley, 1985, p. 85)

As stated before, since 1980 growth in the number of applications puts more emphasis on productivity performance of the computer programmer. Unfortunately for the computer industry, growth in the number of programmers is not keeping up with demand. The growth rate in supply of programmers since 1980 is 15 percent per year while the demand has risen

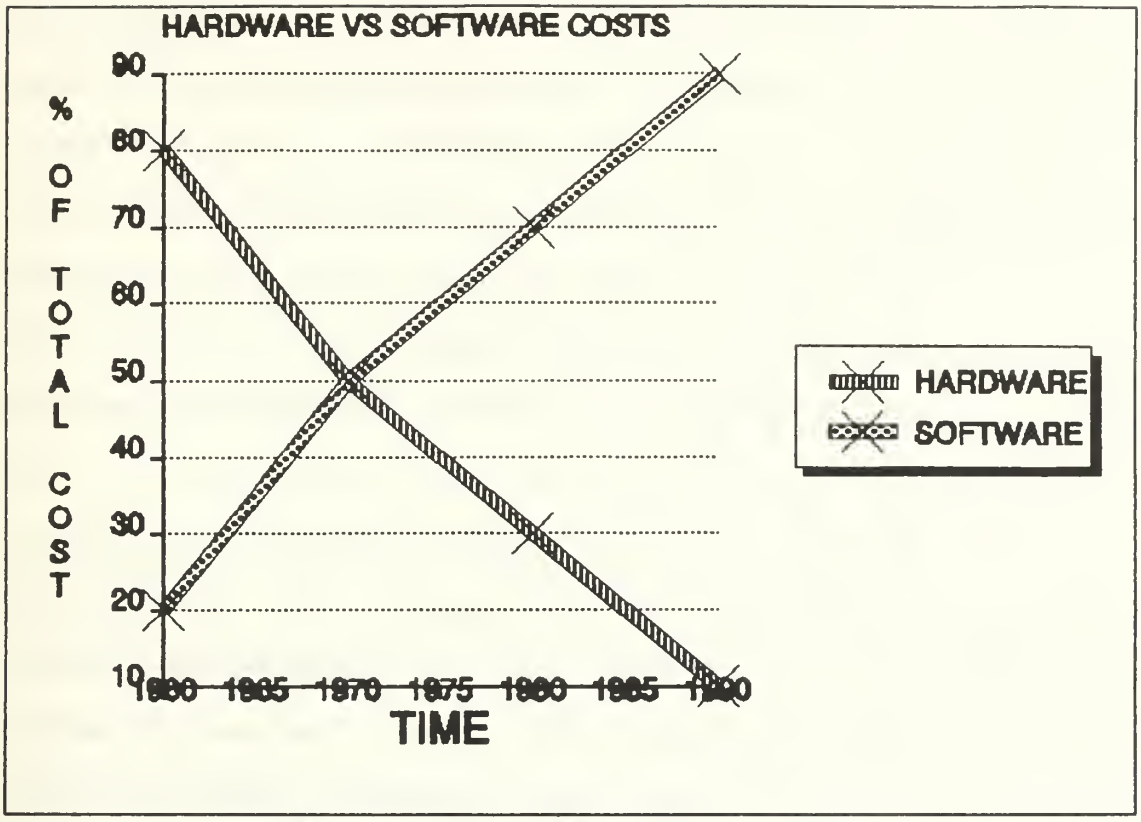

Figure 2 - Hardware vs Software Costs

by 372 percent. In terms of people, this means by 1990 it is estimated the demand for programmers will out pace supply by 750,000 to 2,000,000 personnel (Fairley, 1985, p. 8).

The productivity increases of programmers is also failing to make up for the lack of available programmers. A study conducted from 1973 to 1983 indicated programmer productivity only improved at an average of four percent a year (Martin, 1983, p. 3). So while demand for programmers is increasing, productivity of existing programmers is not keeping pace with growth in software demand.

The characteristics of software itself has also undergone several changes that have contributed to this software crisis. Today, software must be more reliable, easier-to-use and user-friendly than ever before (Conte, 1986, p.2). This reflects increased reliance upon software to run advanced systems and also increased expectations of end users who have become educated in uses of computer technology. Thus, even more pressure is placed upon programmers.

This burden is not solely placed upon the programmers' shoulders. Developmental techniques and productivity enhancement tools have been created that supposedly ease the coding load. These products seek to improve either the coding itself by making it less time consuming (reuse of code, automatic generation of code, fewer lines of code necessary to do the same function, etc.) or less difficult to understand (menu driven formats, non-procedural languages that mimic natural languages, etc.).

2. Software Development Methods

Changing the way software is developed is seen as a possible solution for the software crisis. In the 1960s systematic approaches were created because many of the delivered applications were over budget, behind schedule, inefficient and did not satisfy the original requirements (Fairley, 1985, p. 5). The name "structured methodologies" is given to systematic approaches which have been developed.

Specific names of structured methodologies are many: The Systems Acquisition Process, the Cost Model, the Prototype Life-Cycle Model and the Waterfall model.

Each of these models generally follows the same stages in creating a program. A typical representation of these stages is shown in the System Development Life Cycle (SDLC). In a simplified form, the SDLC steps and questions the steps answer are:

- Recognition of a need: What is the problem?
- Conduct of a feasibility study: What are the facts and user requirements? Is the problem worth solving?
- Design the solution: How must the problem be solved?
- Implement the solution: What is the actual operation?
- Maintain the system: Should the system be modified to change with different user requirements over time?

These steps are conducted by a development team as they study and interact with end users. Thus, the SDLC is a way to bring together two types of computer users: computer programmers and computer application users.

As the steps are followed in a structured approach, the software being developed is supposed to be delivered on schedule, on budget and in a maintainable and acceptable form (Yourdon, 1988, p. 1). But the use of structured development techniques alone can only account for productivity gains up to 25 percent (Martin, 1982, p. 43). To achieve the productivity

gains required to overcome the software crisis, other methods must be used in conjunction with structured approaches.

3. Evolution of Computer Languages

Since the early 1950's when assembly languages were created in order to simplify the task of coding machine language instructions, programmer productivity has been tied to both the ability of the programmer and the language in which coding was conducted. Attempts to increase programmer productivity can be categorized in three areas:

- Further train the programmer. The disadvantage is the time spent in training is lost until the person starts coding again
- Let the person gain experience through time and effort. Productivity differences of up to 20 to 1 have been found between entry level programmers and experienced programmers when using a 3GL (Sackman, 1968, pp 3-11). This method is costly though in the early steps of the learning curve as the beginning programmers learn the language
- Use a more advanced language that facilitates the coding process. The more advanced language must be taught to the programmers.

It is the third option that has received the greatest scrutiny because it is the option that allows greatest potential gains.

This third option has led to the creation of many computer languages. The evolution of computer programming languages is categorized in the form of "generations." In figure 3 is a list of each generation. With each generation

is listed the identifying characteristics and the chief benefits over the previous generation.

GENERATION	CHARACTERISTICS	BENEFITS
First	Hardware Specific.	Formalized programming.
Second	Hardware Specific. Symbolic coding.	Simpler coding. Requires less commands.
Third	Non-Hardware Specific. Requires a compiler.	Simpler coding. Commands are problem orientated vice machine orientated. Requires less commands.
Fourth	Non-procedural.	Easier to learn. Requires less commands. (Awad, pp. 104-114, 1988)

Figure 3 - Computer Language Generations

Studies show productivity improvements of up to 50 percent when a switch is made to a higher programming language. These gains in productivity reflect a 25 percent increase when structured development techniques are used and a gain of another 25 percent when a higher language is also used (Martin, 1982, p. 44). But once again because of the severity of the software crisis, still larger gains in productivity are needed.

C. DEVELOPMENT TOOLS

The results of incorporating development tools do show the productivity gains necessary to combat the software crisis. Productivity gains over 1000 percent are not uncommon with the use of tools (Martin, 1982 p. 44). The types of tools vary greatly but the reasoning behind their creation is the same . An individual tool simplifies an action that is too difficult or tedious for a programmer to do manually (Panko, 1988, p. 17). Without writing a single line of code, a user can create a custom application (Panko, 1988, pp. 456-458). This frees the end user from relying upon a programmer.

As database management becomes the cornerstone of information systems technology (Chorafas, 1986, p. 6) and systems developers attempt to incorporate end users in all aspects of a computer systems development, creation of database applications has become one of the largest areas of programming today. Thus it is not surprising to find some of the greatest emphasis of tools in database management. Structured Query Language (SQL), Query By Example (QBE), dBase, FoxPro and Paradox Application Language (PAL) are a few examples. Each tool allows end users to directly manipulate the database in order to bypass the system analyst and programmer. This bypassing saves time and increases productivity.

Usually a tool follows a series of menu driven steps in which the system structure is developed and the actions to be

taken upon the system are defined. Tools for databases are most frequently used because the actions to be taken on a database are standardized (Input, delete, update, sort and inquiry). Customization takes place as end users design the formats of the output reports and the input screens. Users also will define how they want the data sorted and displayed on the screen. Constraints can be entered to ensure the data entered is within preset limits. Security features such as a password system can be installed and some systems even have features to handle up to Secret level documents that require special erasure routines.

4GL tools generally claim success in two areas: (1) Increased end user programming capability and (2) Increased professional programmer productivity (Chorafas, 1986, p. 3). Inclusion of the end user as the actual programmer allows immediate feedback in meeting demands of the user. Firms which have extensive database dealings often make this switch. In 1981 Xerox Corporation estimated 25 percent of the company's computer resources were dedicated to end user computing and by 1991 this percentage was predicted to triple (Nelson, 1989, p. 1).

The success of database application generators for professional programmers can be linked to several factors. First is that the data types are predefined. The application generator does not have to structure the data or define it. The database itself has already done this for the application

generator. Second, the operations conducted on databases are standard. Data entry, update, deletion, retrieval and inquiry constitute the five standard actions conducted on a database. Third, the outputs from a database are also standardized. They are graphs of individual data fields and summary fields and reports based on the same fields (with the headers already defined). Finally and most importantly to both the programmer and the end user, is the use of the non-procedural 4GLs and menus in the creation of the applications. A high level of training, compared to 3GLs and even 4GLs without a menu driven format is not needed to produce a high quality program in a short period of time.

D. PRODUCTIVITY

1. Definition

What is this productivity that the different languages and tools boast they can improve and thus solve the software crisis? In general terms, productivity is the ratio of what comes out of a process divided by what goes into the process. An increase in productivity would mean either more is being produced with the same level of input or less input produces the same level of output as before. In simplest terms it means getting more for less.

The first step in measuring productivity is to determine units of measurement for both inputs and outputs. If no measurements can be found then no determination of

productivity can be made. Traditionally, in the field of computer software development, inputs have been measured in terms of level of effort (man-months for example) or in terms of dollars. Calculation of the level of effort can lead to some misunderstandings due to confusion of whom is included. Some productivity measurements include support personnel and others include only programmers and analysts. For the purposes of this case study, only programmers are included for two reasons: First, in the SDLC design phase only coding, debugging, and error checking steps were observed. Second, and most importantly, traditional boundaries between roles of programmers and analysts were changed by use of a 4GL application generator thus making any comparison to older studies invalid.

2. Lines of Code Methods

While there is general agreement on measurements of inputs there is general disagreement on measurements of outputs. The most popular, in terms of which have been used the longest and most frequently, are the Lines-Of-Code (LOC) based methods. Foremost among these are the COCOMO (COnstructive COst MOdel) and SLIM models. These methods are easy to understand in concept and the measurements are simple to conduct. They also have a large historical database to draw from and base comparisons on. But, traditionally three areas complicate their use:

- They are not directly transportable across organizations. A calibration process must be accomplished to ensure characteristics of the organization using the model are accounted for before any meaning set of predictions or measurements can be made.

- They are very sensitive to differences in LOC counting methods. The fundamental question of: What is a line of code? must be first answered. Some methods count only executable LOC. Some methods count executable and data definition LOC. Another counts the LOC changed during maintenance only. (Arthur, 1985, pp. 16-28)

- They penalize high level languages. A comparison of the number of LOC needed to produce the same program in terms of functionality between a 3GL and a 4GL will show the 3GL will require seven times more lines of code. (Dreger, 1989, p. 5)

3. Function Point Analysis

An alternative to LOC based methods adopted by the British government and 300 major corporations (Jones, 1991, p. 8) is the Function Point analysis method. This method is based on measuring functionality of the program requested by end users. A function point is defined as one end user business function. The basis of the measurement system is a user must request a function in the specifications in order to include it in the measurement. This requirement is intended to cut down on programmers adding unwanted functions for their convenience. Function points deal with one or more of five general areas:

- Outputs: items of information processed by the computer for end users

- Inquiries: Direct inquiries into a database or master file that look for specific data

- Inputs: Items of data sent by users to the computer for processing and to add, change or delete something
- Files: Data stored for an application, as logically viewed by the users
- Interfaces: Data stored elsewhere by another application but used by the application under evaluation.

In general terms, function point analysis of a system counts the function points and then, by using a series of matrices, determines system complexity . The number of function points and a quantitative expression of the complexity level are then plugged into an equation. This arrives at a number used to predict the level of effort required for system development. As with the COCOMO model, the function point analysis method must be calibrated for the organization's individual characteristics before results will be accurate. (Dreger, 1989, p. 5)

4. Summary and Application

Regardless of which method is used to measure productivity, there are several quality characteristics to be examined in order to assure any comparison of productivity measurements is accurate. The following is a list of aspects to consider in regard to the quality of software:

- Accuracy
- Error tolerances
- Execution efficiency
- Simplicity of the design
- Modularity
- Instrumentation
- Flexibility of the system toward change
- Testability
- Maintainability
- Reliability
- Reusability.

A detailed examination of the applications quality delivered to end users by software development organizations observed in this study is beyond the scope of this paper. All applications examined in this paper have been delivered to end users and are presently fulfilling the tasks they were designed to do. Therefore, accuracy, error tolerances, instrumentation, reliability, testability and simplicity of design are all assumed to be equal among products produced and delivered. The remaining characteristics are all addressed in summary terms vice specific terms related to a particular program. Given the limited number of programs observed in development, extrapolation to any broad conclusion in these areas is unfounded.

For this study, no formal model of productivity was used. The organizations observed measured input in terms of the number of people working on the system times the number of calendar months they worked on it. Output was defined as simply a working system. This was done for several reasons: First, the organizations have no formal method of measuring productivity in place. They have been ordered to institute function point analysis and are in the process of setting it up. Second, no LOC data was kept on any system. Third, the systems observed used different languages and even different generations of languages. Thus any judgement from a comparison of other than total man hours spent developing systems would be unfounded. Problems this represents are discussed in the conclusions section.

E. COST/BENEFIT ANALYSIS

Once the liabilities and strengths of various systems are determined, a way of selecting the best system is necessary. A method for selecting between competing alternatives is by conducting a Cost/Benefit Analysis. A cost/benefit analysis seeks to identify all expenses and advantages associated with one product when it is compared to others. This method is selected when each of the alternatives does not offer the same level of benefits. The costs are normally quantifiable in terms of dollars expended. The benefits though are divided into two categories of quantifiable and nonquantifiable

benefits. Examples of quantifiable benefits are decreased errors per one hundred lines of code, increased number of transactions processed per hour and decreased time to produce a report. Examples of nonquantifiable benefits are increased data availability, increased data timeliness and increased data accuracy.

For each system reviewed the cost/benefit considerations are identified and explained in the last section of each review chapter. The performance costs listed are from functionalities offered by either of the other systems examined that the system in question does not offer.

The overall cost/benefit analysis is conducted for the nine commands who attended the 1991 Ada Technical Workshop. These commands are:

- NCTS San Diego
- NCTS Newport
- NARDAC Norfolk
- NARDAC Washington
- NARDAC Pearl Harbor
- NARDAC Pensacola
- NARDAC New Orleans
- NARDAC San Francisco
- NARDAC Jacksonville.

They also all report to NCTC so uniformity can be controlled by one senior command. These nine commands have all been

directed to implement Ada and also include the two commands that implemented the Off-The-Shelf and In-House developed systems. The costs and benefits for an activity are the average cost for all nine commands except as otherwise noted.

Since the activities all operate under the Navy Industrial Fund concept, all work conducted by the employees must be charged to clients. This includes overhead charges such as training. The fiscal year 1991 rates are as follows:

Title	RATE/HOUR
• Senior Analyst	$43.1481
• Programmer/Analyst	$38.1299
• Junior Programmer	$27.3940
• Admin Support	$29.4167.

These figures are used when calculating opportunity cost of training, cost of product development and cost of product maintenance.

F. RESEARCH QUESTIONS

The following specific research questions will be addressed in order to determine the correct DBMS development tool for the Navy to procure:

- What are the characteristics of a DBMS development tool that contribute to end user and programmer productivity?

- Which method of procurement, In-House, Off-The-Shelf or Contractor aided, should be used to obtain the Navy's DBMS development tool?

- Which of the tools offered by the three procurement methods provides the Navy with the best productivity potential for end users and programmers? Which tool has the greatest potential for future growth?

- How can the Navy best use the services offered by commercial and other DOD sources as it attempts to improve productivity and yet face a decreasing IR budget?

G. THESIS ORGANIZATION

In the following six chapters, three DBMS software development tools are evaluated. Their advantages and disadvantages are discussed in comparison to each other and to an ideal DBMS as defined in Chapter II. Chapter VI contains a cost/benefit analysis of the three tools. The seventh chapter includes results and recommendations from the comparisons and analysis. The recommendations address which software development tool should be selected for the Navy, what that tool should offer and how it should be procured.

II. DEVELOPMENT TOOLS

A. DEVELOPMENT TOOL CHARACTERISTICS

As previously stated, the database is the mainstay of the Information Technology field. In the original conventional file environment each application for a database had a separate set of data elements. But data integration solved this problem and allowed applications from different users to use the same data for separate tasks. This saves data storage costs, reduces data redundancy and improves data integrity. Handling of data integration is the heart of the database system. A Database Management System (DBMS) accomplishes this task.

The basic functions a DBMS provides are:

- Establishes data relationships within the system
- Allows users to add new information, change information already loaded, delete old information, sort information into a useful order, and search for particular types of information
- Controls concurrent processing by two or more users on the same data
- Provides security to identify users and authorize actions
- Provide facilities for recovery of the database from system failures. (Simpson, 1989, p. 8)

Since actions performed on a database are standardized, the area of database management programming shows great potential for end users to create their own applications. End users know what they want from their database applications, so want they need is a tool to free them from reliance upon system analysts and programmers to create their applications. Data organization is familiar to end users. The conceptual workings of a database (Update, entry, deletion, query, sort) are also familiar.

This is the reason behind creation of automatic code or application generator tools. These products follow a menu driven format that takes end users through a series of steps that define data elements, establish relations between data elements, create data entry screens, and format output reports. The menu concludes with the code being automatically generated with inclusion of standard database functions. The result is creation of a stand-alone database system defined and created by end users without the aid of system developers or programmers.

A questionnaire was distributed to both end users and programmers of database systems. They were asked to rate characteristics they desired in a database system. The scale was from one being the most desirable trait to 13 being the least desirable trait. All end users lacked previous formal computer training in either programming or computer systems. The professional programmers have at least a bachelor of

science degree in computer science and on average five years of programming experience on database systems using both 3GLs and 4GLs. Results are shown in Figure 4.

	Programmers	End users
System response time	2	4
Ad hoc capability	8(tie)	2
Ease of use	4	1
Ease of learning	1	5(tie)
Mouse compatibility	13	13
Graphics	10	8
Documentation	6	5(tie)
On-line help	7	9
Training from the vendor	12	7
Maintenance ease	8(tie)	11
Security features	11	10
Data integrity	3	12
Reliability	5	3

Figure 4 - DBMS Characteristics Ratings

Two areas with a high differential between end users and programmers preferences, data integrity and training from the vendor, are each caused by a temporary condition. First, there was a misunderstanding in terms used on the questionnaire and second, context of the workers' jobs when they answered the questionnaire influenced their answers.

The term "Data Integrity" was misunderstood by all end users who took the questionnaire. Their perception about its meaning ranged from "I had no idea so I ranked it low." to "I thought it had to do with how the data was read off of messages when it was input into the system." When it was

explained after finishing the questionnaire, each answerer said they would have ranked it higher had they known what it meant.

The difference in ranking of "Training from the vendor" is explained by understanding the stage users were in when they answered the questionnaire. The users had just switched from an old system that offered little flexibility but they understood how it worked. The new system offered the flexibility and functionality they needed but training was required so they could operate their system to it's full extent. Users saw "Training from the vendor" as an immediate solution to their learning curve problems. The programmers rated this characteristic from a life cycle perspective. They thought it was important when the training occurred but throughout the life cycle it did not rank highly.

Other than these two areas, ad hoc capability stands alone as the one area where end users and programmers disagreed. This is not explained by a temporary condition but by a difference in development philosophy. In fact, if the programmers who did supply ad hoc capabilities to their end users were factored out, the difference becomes even larger. It must also be noted if the end users who presently did not have ad hoc capabilities rated this characteristic more in line with the programmers.

Senior analyst personnel also exhibit a difference in opinion with users over desirability of ad hoc capabilities.

In interviews with the author, systems analysts for DB3GEN at the Naval Regional Data Automation Center (NARDAC) San Diego and Clipper at NARDAC Norfolk stated 95 percent of all end users do not want ad hoc capabilities. This means developers do not think end users want ad hoc capabilities so they create applications without them. But end users with the capabilities not only desire them but rate them second highest in desirability out of all characteristics. The results indicate the users cannot want something they do not have.

Besides these three mentioned traits, end users and programmers were in general agreement over the importance of the characteristics they wanted in a database system. Ease of use, ease of learning and system response time were at the top of both lists while mouse compatibility was at the bottom of both lists. As a note, no responders to the survey were presently using a mouse on their computer.

B. SURVEY RESULTS

Once determining what the users and programmers wanted in a system, another questionnaire was administered to measure the satisfaction they had with their present systems. The following twelve areas are traditionally used to measure user satisfaction of a Full-Featured MS-DOS DBMS (Microsoft-Disk Operating System Database Management System) (Robb, 1990, p. 12).

Results are as follows in Figure 5: (1 = excellent, 2 = good, 3 = fair and 4 = poor. An asterisk (*) indicates no evaluation or not known.)

	DB3GEN	PARADOX	ADASAGE
Reliability	1	1	1
Documentation	4	4	3
Ease of use	2	1	2
Report Generation	2	1	2
User Interface	2	1	1
Standards	*	2	1
Programming Features	2	1	1
Speed	1	2	1
On-Line Help	3	1	2
Vendor Support	1	1	1
Value	1	2	1
Installation	*	1	4
Average	1.9	1.5	1.7

Figure 5 - Survey Results

On the average, the Paradox system scored better than DB3GEN and AdaSAGE. All three systems scored, on the average between excellent and good in overall rating. This makes sense when considering each person polled, except for four end users, had selected their present system over competing systems. This poll does not show a preference of users and programmers of one product over another. No user or programmer of any system had used either of the two other systems. This poll does show the Paradox programmers and end

users are slightly more satisfied with their product than the other two systems.

1. Documentation

A constant discrepancy between all three systems is in the accompanying documentation. Reasons for this discrepancy for each systems varied.

The DB3GEN system required end users to print out the documentation from computer disks supplied with the program. The print out is hard to read as a stand alone document. Users must follow the program on the screen or the documentation does not make sense. As users create systems, the documentation does not explain the reasoning behind the actions you are required to make. This is due to users of the DB3GEN system are normally programmers or advanced users who are already trained on the system. No system is delivered without first training end users on the system and most applications are created by the programmers. Training of users on the system accomplishes feedback but the application is delivered as a finished product without the complete ad hoc capabilities of a Paradox system. The senior DB3GEN analyst says 95 percent of the end users who have systems delivered to them do not want ad hoc capabilities. If this is true for their end users then, the documentation discrepancies will not affect the system performance because the users will not need it.

The AdaSAGE documentation scored the highest of the three but it still is poor, on the average, according to end users and programmers. Documentation is also one area requiring improvement as identified by the Ada Technical Workshop. From interviews with AdaSAGE programmers in the United States Marine Corps, 25 of the 37 procedures in the AdaSAGE library are not used. Of those 25 unused packages, seven are not used because programmers do not know what the package does or cannot figure it out from the documentation. It must be noted non-use of these 25 procedures has not caused failure of any application to be developed.

The Paradox documentation problem was due to the rapid system development time. The programmers were able to deliver the application to the end users faster than they had planned so the documentation was not ready yet. The situation was made worse by users needing more documentation than with a non-ad hoc capable system. The ad hoc capabilities require more user involvement. This increased involvement led to more user questions. More questions meant more reliance on documentation and it was not ready. The system did have on-line help features but traditional reliance upon printed material for help caused end users to call the developers for help frequently. (As often as twice a day when the system was first delivered.) This further delayed the documentation by slowing down it's creation. One user was so frustrated, he

went out on the commercial market and bought a manual with his own personal funds.

2. System Speed

The speed scores were excellent for the DB3GEN and AdaSAGE systems and good for the Paradox system. Ordinarily this low difference in opinion should not be an area of concern but it is since speed was rated high in importance of characteristics by programmers and end users. Speed is a selling point of AdaSAGE according to documentation accompanying the AdaSAGE demonstration disks.

Several factors can affect system operation speed. First, is the system hardware. Processors operate at different speeds so applications have different run-times on different computers. What takes an unacceptable length of time for an operation on an older 286 based PC might be acceptable on a newer 386 based PC. Computers with the same processor can even operate at different speeds. Addition of extra cache memory or a co-processor will speed up execution of a program. Thus program operation time may be unacceptable only on certain computers which have certain hardware configurations.

Another consideration, especially for the AdaSAGE system, is which compiler is used. Paradox and DB3GEN both always use their same compilers so their inputs are constant. The AdaSAGE program presently only uses one compiler, the

Alsys compiler, but other compilers may soon be on the market. As stated in the AdaSAGE response testing section, the M2SAGE performed faster because it used a faster compiler. As more compilers enter the marketplace, AdaSAGE users should be aware of the affects of this component on system performance.

The way a program handles memory as it executes is another factor affecting system operation speed. The more random-access-memory (RAM) left available for the program to conduct its operations and the fewer times a program must go to disk storage memory, the faster it will operate. The way a program is written and its size will determine how it operates.

The Paradox program includes operations users may not use each time they execute the program. These unused operations still require added program code that must be loaded into RAM and then executed. The program has to check and see if the user wants to use an operation regardless of whether or not it is to be used. The DB3GEN and AdaSAGE programs only offer end users the functionality they request (and the system can offer). This means the application does not have to query the system to find out if the end user wants to do a specific operation except at the menu screens. And even at the menu screens there are fewer options afforded users that corresponds to less code to load and execute. All this adds up to faster times as evidenced by the speed tests in the earlier section.

Developers at the Norfolk Naval Regional Data Automation Center (NARDAC) have gone one step further in order to speed up Paradox developed systems they deliver. The slow part of the process was the message loading section. This section was rewritten instead in the C language and the rest was written using the Paradox Application Language (PAL). This decreased message loading time by half. The program was not tested using the new 386 based desktop III workstation with the extended memory or with the updated version of the Paradox system. Either of these new options should decrease the program run time.

C. RESULTS

The one common trait between the two questionnaires is users and programmers are content with what they have. This means they are either satisfied with what they have or, as with the first survey and ad hoc capabilities, do not know what they are missing.

Users and programmers agree they want systems to be fast to operate, easy to use, and reliable. Users also want better documentation, training from the vendor and ad hoc capabilities. Documentation and training can be improved readily and without any fundamental changes to the system. Ad hoc capabilities are different. They represent a basic difference in how the system develops applications and how the applications operate.

To analyze the difference between having and not having this capability, the costs and benefits of the three products must be found and compared. This is done in the following chapters.

III. OFF-THE-SHELF: PARADOX

A. OVERVIEW

The Naval Regional Data Automation Processing Center (NARDAC) located in Norfolk, Virginia creates programs for Department of Defense (DOD) commands under the Industrial Fund concept. This concept requires the NARDAC to compete as a self-funded organization for Information Resources (IR) dollars of DOD activities. This is in direct competition against commercial organizations and other NARDACs for contracts. When NARDAC Norfolk is awarded a contract to create a computer system or a computer application for a specific dollar amount, the NARDAC acts just as a commercial corporation would: To maximize the output of the employees while minimizing the expenses incurred. This ratio is the productivity of the organization.

B. THE TYPE COMMANDER HEADQUARTERS AUTOMATED INFORMATION SYSTEM (THAIS)

In 1979 Norfolk NARDAC was designated the Central Design Agency (CDA) for the Type Commander Headquarters Automated Information System (THAIS) contract. The THAIS mission is to provide an on-line, interactive management information system to seven Type Commanders (TYCOMs). It creates a centralized database in 10 functional areas to provide earlier problem

recognition and expand data utility to more end users. Through use of a hierarchial structured database, the on-line portion is menu driven, provides summary data for staff analysis, and produces standard reports and ad hoc queries. The system was originally designed to be run on SNAP I hardware consisting of three Honeywell DPS-6 mini-computers. No microcomputer interfaces were in the initial design.

The original system was written in COBOL and delivered to the TYCOMs. The system had several stumbling blocks that limited use of information in the database by end users. Primary of these stumbling blocks were time delays in creation of reports involving graphics and lack of capability to create ad hoc reports and forms. Neither of these two areas were requested in the original specifications but their need arose after delivery. The cost of rewriting the existing system to include these capabilities was prohibitive (two to three months of effort by four to six programmers) so manual methods were found to make up for lacking functionality.

Two reports that could not be created with the original system are the OPTAR (Operating Target: a financial budgeting report) and the Performance Summary report. Each report requires addition of data fields that are not in the original database and summary totals of old and new data fields. In order to produce the new reports, an operator down loaded data in the database, had a second operator add two fields of data, and then process the data on a 286 based Personal Computer

(PC) using the LOTUS 1-2-3 program. This entire process took three weeks.

To create the graphics reports, data from the database was also down loaded from the Honeywell minicomputer and input onto a 286 based PC. This process took the system operator from one to two weeks depending on the amount of data. This meant the information was at least one week old (and at most three weeks) when the report was finally printed. During that time the system operator could handle any new requests or correct mis-entered data.

C. NEW SYSTEM DEVELOPMENT AND IMPLEMENTATION

The senior analyst for the THAIS contract (and also an experienced COBOL programmer) was concerned with the system's inability to provide end users with the desired functionality. The analyst also realized system maintenance was causing too great a workload on his programmers. He realized any future module designed, coded, and delivered in the same manner, would also contribute to the maintenance backlog.

After a review of maintenance requests submitted by end users, the analyst determined a majority of requests asked for reformats of outputs, additions to outputs or added functionality. To identify existing system deficiencies, the analyst interviewed end users and arrived at this list of needed capabilities:

- Ad hoc report generation
- Ad hoc form generation
- Graphics
- Security (to handle secret level material).

With these requirements in mind, plus the original database specifications, the analyst set out to find a 4GL and/or database package to solve the problems. The system selected was the Paradox 3.0 database management system.

Paradox 3.0 system is described by its manufacturer as a multi-user, relational database compatible with the industry standard dBase series data file format. Queries are conducted by the Query-By-Example (QBE) method instead of the Structured Query Language (SQL) and programming is done either manually using the Paradox Application Language (PAL) or automatically by the application generator. The system allows the user to modify the pop-up boxes and screens instead of just presenting them in one format. This is accomplished either through use of a keyboard or a mouse. The program also includes extensive on-line help features which assist end users in all phases of application development and use.

Other systems on the market provide similar functionality but the decision to select the Paradox system was based on three factors: First, it (Paradox 3.0) met all requirements; Second, it was within the price range of the buyer and third, it was the first system tested that met all requirements.

Two of the original THIAS programmers were selected to create the new database system using Paradox. Neither programmer had ever coded using either a 4GL or an automatic code generator. Including time spent on learning the new program, the two programmers took only two months to produce what had taken six experienced programmers six months.

D. RESULTS AND FEEDBACK

Once the new system was delivered, improvements were realized immediately. Time to create ad hoc reports fell from three weeks to one day with the limiting factor being print time. Time to create graphical reports fell from one week to also one day. An unanticipated area of improvement was in loading or "reading" of messages containing data into the system. The previous system had a ten percent error rate. These errors were manually corrected by the system operator. The new system has yet to have an error in data input from messages thus freeing up 10 man hours per week.

Another area of increased benefits is the timeliness of reports. Under the old system, the reports usefulness was limited to tracking trends over a long period because the data was at least two weeks old. But since the new reports were only one day time late because of the ad hoc capability they are now used to monitor day to day progress of fleet activities.

The ad hoc functionality also improved the readability of the reports. Originally, the reports came in only one form so end users had to glean needed information out of all the data presented. Ad hoc capabilities allowed users to modify reports to suit the readers. The increased readability and timeliness of the reports has led to a doubling of requests for the reports since implementation of the new system.

Two areas of concern have been raised by the switch to the PC based Paradox system. First, periodic reports require 400 to 700 pages of output and the microcomputer's printer will not handle such a load. This problem has been solved by loading report data to a mini-computer to print the report. A long term solution is presently being sought.

The second area is decreased turn around time in delivery of end products to users. The rapid development time, leaves programmers with less time to produce documentation. This is a management issue and does not reflect on any discrepancy with the system.

IV. IN-HOUSE DEVELOPED: DB3GEN

A. BACKGROUND

In 1981 Naval Computer and Telecommunications Station (NCTS)(Formerly the Navy Regional Data Automation Center (NARDAC)) San Diego, introduced DB2GEN. It allowed professional programmers to create applications rapidly for end users with little training in dBase programming. It also alleviated advanced end users from having to rely on professional programmers to customize or alter a program in order to conduct ad hoc queries.

B. SYSTEM DESCRIPTION

DB3GEN, the updated version of DB2GEN, is similar in design to commercially available products. It, like other DBMSs, focuses on the standard operations end users conduct on a database. Once preliminaries such as naming the system, entering the date and creating a password are done, the system creator lists the field names in the data file. (The data file must be created separately.) From these files "quick access" fields may be selected. These are the fields by which the user desires to call up or retrieve records. Sorting or conditional information may be entered at this time when the user wants to sequence on multiple fields. The creator also

may validate fields (within specific limits) for correct data entry.

Standard operations which all DB3GEN applications have are:

- Add
- Change
- Delete
- Display
- Reports
- Ad Hoc Query
- System Utilities
- Change to Another System
- Exit the System.

Any other operations that users require must be manually coded by the system creator.

Working with the industry standard dBase file structure, DB3GEN allows programmers to change the manner in which data is sorted, queried, displayed and updated. Turnaround time is now hours (however long it took the user to change the system) instead of days or weeks when programmers have to change the original source code. This automated capability, as with all development tools, means increased productivity.

The problem with DB3GEN comes not from the program itself but from a change in how business is to be conducted at all NARDACs across the country. In the early 1980's, in an effort

to curb the rising cost of Information Resources (IR), NARDACs and NCTSs were changed from being independently funded to organizations funded under the Naval Industrial Funding (NIF) concept. This means any system development at a NIF activity has to be funded by a DOD client. DB3GEN did not fall into that category. Even though it was designed to fit a need and increase productivity, lack of a sponsoring (funding) organization has frozen the status of the system. DB3GEN can be used to develop applications but no improvements to the system can be made by NCTS San Diego programmers.

Presently, DB3GEN does not offer complete ad hoc capability to end users; especially users who are unfamiliar with dBase III+ programming commands. The applications it creates are easy to use and understand but they cannot be altered without changing the entire program.

All official work on improving and modernizing DB3GEN ended with implementation of the NIF. Since then, commercial products have been introduced on to the market that not only perform the functions DB3GEN but add to them. Examples include dBase IV, Foxpro and Paradox. Needed changes which programmers have been unable to make include addition of pop up windows, graphics capability and creation of a Local Area Network (LAN) version.

V. COMBINED MILITARY AND CONTRACTOR ASSETS: ADASAGE

A. BACKGROUND

Ada is the only major language developed primarily for modern embedded processors (Johnson, 1985, p. 1). But it is the Department of Defense's (DOD) decision to adopt it as it's standard computer language that makes Ada unique among all languages. The primary reason to standardize is to reduce life cycle costs. In the late 1960's DOD, as it tried to update the World Wide Military Command and Control System, found it was faced with the task of integrating over 800 applications written in over 30 different languages (Schatz, 1985, p. 22). Ada was adopted in order to simplify this problem. Testifying before the House Committee on Science and Technology, Dr. Edith W. Martin, the Deputy Under Secretary of Defense for Research and Advanced Technology, stated "The potential benefits due to the use of Ada have been estimated, via three independent studies, to be in excess of $1000 million per year." (Johnson, 1985, p. 22)

Ada was not the first language developed by DOD. COBOL was designed over thirty years ago to fill much the same need Ada faces today. Technical merits aside, the requirement by DOD for use of COBOL in all defense computer contracts made it the most popular programming language in the government and

private industry. This created economic demand and private industry followed suit. It is estimated the market for Ada language programs alone in 1985 was $6 billion. And with the need for software in DOD rising at a rate of 12 percent a year (Johnson, 1985, p. 2), this equates to a demand of $11.8 billion today.

In 1974, twenty-six different languages were initially evaluated to become the DOD-wide standard but all were found to be inadequate for the tasks required. Then in 1977, in response to the problems identified in an earlier study, two Pascal based languages were developed and combined to form the language then know as DOD-1. This language was renamed as Ada. The calibration process for requirements and standards for the language were further updated until 1983 when the DOD and the American National Standards Institute (ANSI) released the *Reference Manual for the Ada Programming Language, ANSI/MIL-STD-1815A*. This document continues to be the guiding standard today. (Wallace, 1986, p. 6)

There are three guidelines that require use of the Ada language in the DOD. They are:

- DOD Directive 3405.2 of 30 March, 1987
- DOD Software Master Plan of 4 June, 1990
- Section 8084 of the 101st Congress House Appropriations Bill.

All three address the use of Ada in the DOD in broad and specific terms. The later two documents address the inclusion of Ada in non-tactical software development where before only mission critical software had to use the Ada language.

The three documents do not specify how the transition will be funded and this is very important in the Navy Industrial Fund (NIF) environment. Since end users specify what products they want and the activity must create the products as inexpensively as possible, decreased productivity during the learning curve process is a detriment to any change not accompanied with funds to assist the switch.

B. DEVELOPMENT ENVIRONMENT

By design, Ada is intended to support modern software engineering concepts and principles. This means the language requires an automated environment. In the environment, software development tools must be able to develop, support and maintain software over the entire program life cycle. The environment presently being used by the DOD is the AdaSAGE system. This environment is currently provided under a Department of Energy (DOE) contract to DOD activities.

AdaSAGE is an unlicensed public domain product available through the National Energy Software Center at Argonne, Illinois. It is considered a "shareware" program since it was developed with United States' government funds and is available in accordance with the Freedom of Information Act.

It is defined as an Applications Development System for Ada (Taylor, 1990, p. 1). This means it provides an entire development team (including end users) the environment in which a set of utilities can be used to facilitate rapid, end user specific system creation. The utilities are a combination of precoded applications and program development programs. Precoded applications are divided into five general packages: Database, Windows, Communications, Graphics and Documentation. In each package or library there are precoded modules that can be linked together to provide capabilities to end users such as database storage and retrieval, online help, sorting and editing. The linking of modules can be done by either manual coding or by precoded "linker" programs. An example, "THOR", is used to create the data dictionary. By creating a data dictionary the developer defines field names and their relationships; field formats, types, ranges, and valid entries; schema structures; views; passwords; report formats; forms; menus and windows. This is all done without requiring manual coding.

The AdaSAGE environment requires a full function Ada compiler in order to create applications. The compiler of choice among DOD activities is the Alsys 286 compiler. All four DOD activities interviewed that were using the AdaSAGE system, were also using the Alsys compiler. Presently no other validated personal computer (PC) based Ada compiler fully supports the full memory model required by AdaSAGE.

The AdaSAGE program and the Alsys compiler it must run on have memory requirements that must be met before the program can be loaded. A 286-based Personal Computer (PC) system, in addition to the standard one megabyte of RAM already installed, requires at least two more megabytes of RAM and seven megabytes of hard disk space. All United States government PCs have the hard disk space available but no 286 based PC has the RAM space without modification. Most 386 based PCs do have the RAM requirements but the addition of six megabytes of RAM is desired in order to increase the system's performance. For the purposes of this paper only the minimum hardware configuration will be used.

C. REUSABLE ADA LIBRARIES

Individual libraries have various costs for both the installation, training and maintenance of the system. The United States Army has developed the Reusable Ada Products for Information Systems Development (RAPID) system as a comprehensive reuse environment. This system not only centrally locates procedures for other activities to reuse but through design guidelines, methodologies and adherence to strict quality control standards ensures all procedures are compatible with each other. Before a procedure is admitted to the library for reuse, it must meet specific guidelines to the definition of what the procedure does, documentation of how the procedure operates and technical aspects in regards to

variable handling and input\output requirements. By requiring up-front conformance to standards, the RAPID system ensures reusability of procedures by other activities. Strict adherence to documentation standards ensures modules are able to be reused. Programmers can read the model documentation and fully understand what it does and what it needs in order to function.

Ideally one central library could be created but the lack of joint funds for staffing of personnel to maintain and manage the library is a limiting aspect.

The minimum equipment required to run RAPID is extensive. The hardware required is:

- DEC MicroVAX II or
- DEC MicroVAX 3100
- 16 megabytes of Random Access Memory (RAM)
- 1 Console terminal printer
- 7 Terminal ports
- 1 Printer port
- 2 RD54 Disk drives
- 2 TK50 Streaming tape drive
- 1 2400 Baud modem
- 1 VT220 Video terminal
- 1 VT340 Color video/graphics terminal.

The software used by the RAPID system is:

- VMS 5.3
- Ada V1.4
- RAPID Center Library System 3.1
- Oracle V5.1.22. (Piper, 1991, p. 2-1)

The Army's RAPID system is mentioned not as just a possible source of reusable code but as a model of how DOD should design it's own library of procedures. There are presently over 24 different sources of reusable Ada code in the United States (Levine, 1990, pp. 62-65). The RAPID system has the advantage of being a military source plus already having strict guidelines and standards in place. Once the Navy develops its own procedures, they should be placed in this library. This would be the first major step toward developing a DOD-wide non-tactical repository of Ada code.

In order to make the RAPID system a DOD-wide library, the Army would require an increase in its manning level of quality assurance personnel. The Ada Joint Project Office, as the leader of the joint service Ada effort, should select programmers from each service to make this effort truly DOD-wide. It is a waste of resources if each service sets up its own library with redundant staff and hardware. These duplicated libraries would also have to employ their own quality assurance teams to examine software produced, publish instructions on how software should be designed to ensure compatibility and, in essence, recreate the entire system the

Army already has in place. This defeats the very nature of the Ada language: reuse. A DOD library (The RAPID system) saves overhead costs, ensures a DOD-wide standard for Ada software and uses an in-place system so one does not have to be created.

D. FUNCTIONALITY

Presently, the AdaSAGE system is trying to catch up to commercially available database management systems (DBMS). The creators and maintainers of AdaSAGE are quick to point out their system is just not a DBMS. But it is in this area that most of the non-tactical applications and comparisons are made. The biggest discrepancy between AdaSAGE and commercial DBMSs is the lack of end user ad hoc capability in generation and customization of reports, forms, graphics and queries. This problem is known to developers of AdaSAGE and projects such as the QBE procedure development address this issue. The fundamental problem AdaSAGE is faced with is to find funding from governmental activities to develop procedures that commercially products already have. It is against DOD policy to compete with off-the-shelf products according to OMB circular A-76 but it is also DOD policy to support use of the Ada language. So far AdaSAGE is the only development tool which produces it's code in Ada so it is the adopted tool.

An attempt to use the functionality of commercially available programs, such as Paradox DBMS, within the Ada

environment has led to the creation of "binder" programs. A binder allows two programs, written in different languages, to interact. The "binder" acts as a translator between the programs. Since the successful creation of a Paradox/Ada binder by the programmers at NARDAC Norfolk, there is now a tie between Ada and Paradox.

The Paradox/Ada binder is written in a generic format so any database system might be used instead of just Paradox. This does not however give AdaSAGE all the capabilities of the Paradox system. This is because of two reasons. First the binder links only the Ada language and Paradox together and not AdaSAGE and Paradox. A procedure could be written to call a Paradox-like database system in AdaSAGE through the use of this binder but one has not yet been coded. The second and most important reason is that even with the binder, the "bound" program will not produce Ada code. The code produced will remain in the language it was designed to create. The "bound" will still perform as before but its' results cannot be integrated into the AdaSAGE system.

The binder program can be used though in an office management system written in Ada. If personnel are already trained in the use of Paradox or another program and do not want to switch systems, they will not have to with the binder. The binder does not help the move to AdaSAGE but it does help switch to the Ada language.

E. PERFORMANCE

1. End User Considerations

The Marine Corps Logistics Depot at Albany, Georgia conducted performance tests on a database with five different database management systems (DBMSs). The test platform was a Compaq portable 20 Mega-Hertz (MHz) 386 with 2048 kilobytes cache memory, 10 Mega-Byte (MB) RAM and a 110MB hard disk drive. All software programs were installed according to manufacturer specifications. Database on which the operations were conducted consisted of 10,000 records and four fields within each record. The schema definition was as follows:

Field Name	Field Type	Field Length	Key
NUM	NUMBER	5	Primary
ALPHA	CHARACTER	10	Alternate
BIGNUM	NUMBER	8	Alternate
NAME	CHARACTER	20	None

The actions conducted on the database were standard operations for a database: Load, Update, Unload and Delete. The results are shown below:

- Test 1 : LOAD
  10,000 records from an ASCII file were loaded into a database. Indexes were built during or after the operation as the program required.

Program	Time (sec)	Disk space required (Bytes)
M2SAGE	138.25	672,050
AdaSAGE	213.72	672,050
Paradox 3.0	485.88	859,530
Informix 2.10	1384.26	798,720
Oracle 5.1	392.65	517,120

- Test 2 : UPDATE
  An update consisting of subtracting one from the value of BIGNUM for each record was conducted from each record.

Program	Time (sec)
M2SAGE	119.46
AdaSAGE	176.70
Paradox 3.0	1861.00
Informix 2.10	560.00
Oracle 5.1	874.00

- Test 3 : UNLOAD[1]
  All 10,000 records were unloaded into an ASCII file.

Program	Time (Sorted)(sec)	Time (Unsorted)(sec)
M2SAGE	33.12	11.53
AdaSAGE	44.54	17.30
Paradox 3.0	73.07	Not Applicable
Informix 2.10	Not Applicable	42.45
Oracle 5.1	Not Applicable	84.00

- Test 4 : DELETE
  All records that have a value of "P" in the first value of the ALPHA field were deleted. A total of 385 records were deleted.

Program	Time (sec)
M2SAGE	39.16
AdaSAGE	54.60
Paradox	30.00
Informix 2.10	67.00
Oracle 5.1	58.00.

---

[1] The Not Applicable portions are cannot be performed. Paradox stores fields sorted and Oracle and Informix do not. Therefore, unsorted and sorted unload tests cannot be run on each system.

In the above tests, the top performer was the M2SAGE system at 329.99 seconds. AdaSAGE was second with 489.56 seconds and Paradox was third with 2449.95 seconds. The M2SAGE system uses the Stony Brook compiler, which while it is faster, it does not include all of the functionality required under the 1815A mil-spec.

2. Programmer Considerations

Developing applications with AdaSAGE presents several difficulties to programmers. The first problem is a slow compile time. This problem stems not from the AdaSAGE program but from the compiler the system must use. Since the Alsys compiler is presently the only Ada compiler that is both PC based and validated in accordance with the American National Standards Institute (ANSI) *Reference Manual for the Ada Programming Language, ANSI/MIL-STD-1815A*, there is nothing to compare it against except the experience of programmers who have used more mature compilers from other languages such as C, C+, COBOL and the dBase series. All programmers who had coded in another language before switching to Ada and AdaSAGE said the compile time of AdaSAGE seemed excessive or as one programmer said "When you compile, it is a good time to take a coffee break."

The second problem programmers face with use of the AdaSAGE system is in the error handling process. AdaSAGE will cause the system to "crash" when it encounters errors rather

than let the programmer, through error handling routines, account for identification and correction of the error. This is a technical problem that will not be encountered by end users but is very frustrating to programmers as they attempt to debug an application they are creating.

3. Software Development Considerations

The following statistics on Ada productivity come from a study conducted on 75 completed projects consisting of over 30 million lines of code (LOC) from 15 United States' firms. The average project size was 100,000 LOC and the range was from 10,000 LOC to five million LOC. The comparative database of non-Ada projects was from over 1500 projects completed over ten years. The study itself took over three years to conduct.

As the transition to Ada occurs there will be three different cost phases the changing organization will go through. The first phase will see an increase of 10 to 20 percent in the total development cost of a project. This will last for two or three projects until the break-even point can be reached. The development time will take three to five projects before reaching the schedule time break-even point. Increases in cost and schedule time are due to the learning curve of employees as they get used to the reuse environment and structured environment of the Ada language. As a general

rule, there must be at least a 10 percent reusage rate for the project to break even in cost. (Reifer, 1990, p. 475)

After the initial transition period and up to the three year period the average increase in productivity was 20 percent. The cost was decreased 25 to 30 percent; especially when the reusage rate reached more than 20 percent. The degree of planned reusage (as opposed to ad hoc reusage) averaged 18 percent after the transition. The changes in productivity are summarized in Figure 6. (Reifer, 1990, p. 475)

Number of completed projects	Change in productivity	Change in cost
zero to three	ten to 20 percent decrease	ten to 20 percent increase
four to five	none	none
more than five	20 percent increase	25 to 30 percent decrease

Figure 6 - Ada Productivity Changes

The use of a development tool can further increase productivity. Gains as much as 17 percent were found when the tool was integrated to object orientated methods. The average gain in productivity with a development tool was 10 to 12 percent in the study. (Reifer, 1990, p.475)

AdaSAGE, which supports object-oriented design, has been found to have a 65 to 70 percent reusage rate for

Management Information Systems (MIS) applications. This percentage drops to 50 percent if you do not include a second call to a procedure as reuse. This is a direct cause of duplication of applications that MIS systems have at two or more activities. Database management system (DBMS) functions wanted by one command are normally the same functions desired by all other commands. It is this standardization the AdaSAGE system hopes to take advantage of through its reusability of procedures. (Stewart, 1991)

The use of previously coded packages and following the structured format of the Ada language rewards the developer with a decrease in maintenance costs of 10 to 20 percent. The number of errors through out the life cycle is also decreased at a rate of 20 to 30 percent. (Reifer, 1990, pp. 473-475)

VI. COST/BENEFIT ANALYSIS

A. DB3GEN

1. Financial Costs

DB3GEN is free to the programmers of Naval Computer and Telecommunications Station (NCTS) San Diego. It was developed in-house and the costs to create it are now sunk costs. If it were distributed Navy-wide, the cost would be determined by the cost to maintain and update the system. From interviews with senior system developers and programmers, it was determined the staff personnel for the development and maintenance teams would consist of the following:

- one senior analyst
- one programmer/analyst
- one junior programmer
- one administrative support member.

This team composition is for each team for a total of eight employees.

The cost of the teams is based on the following:

- 40 hour work week
- 52 weeks per year
- $276.1774 per hour per team opportunity cost
- Two teams.

multiplying out these three figures arrives at the product of $374,448.00 per year.

The training cost for programmers would consist of one week of instruction on the dBase III+ language and one week on the use of DBICEM. Given the same hourly rates and the substitution of a programmer instructor for the team for the administrative support personnel, the cost of training is determined as follows:

- Two weeks training time
- $146.8018 per hour per team opportunity cost
- 40 hours per week
- Two teams of personnel.

The product of these figures is $23,488.30. The training would have to be conducted on an annual basis to account for the turnover of personnel. A summary of all costs is in Figure 7.

The DBICEM program is compatible with all IBM Personal Computers (PCs) and no extra hardware is required to run the program on personal computers (PCs) presently supplied by the Department of Defense (DOD).

2. Performance Liabilities

DBICEM does not provide graphics, mouse support, secret level security functions, user changeable screens, complete ad hoc capabilities for reports, screens, sorting,

forms and queries or produce its code in Ada. No benchmark speed tests are available using the DB3GEN program or any application it has developed. No library of reusable code exists. All development costs to this day are also sunk costs.

Training Cost

Two weeks * 40 hours/week * $146.80/hour * two teams = $23,488

Development and Maintenance Costs

52 weeks/year * 40 hours/week * $276.18/hour * two teams = $574,448

Total Costs

$23,488.30 + $574,448.99 = $597,937.29 for the first year.

$574,448.99 for each year thereafter.

Figure 7 - DB3GEN Costs

B. PARADOX DATABASE MANAGEMENT SYSTEM

Off-The-Shelf products have an advantage in a cost benefit analysis when compared to In-house or contractor developed products. The purchase price of the development tool includes the development cost; no development team needs to be staffed. The large volume of Off-The-Shelf product sales spread out the costs of development. The other two options

require the full development cost to be paid by the developer. In the case of DB3GEN and AdaSAGE, the developer is the U.S. Navy.

Another advantage Off-The-Shelf products have is they are designed to meet market requirements, thus they are designed to run on computers presently in use. This means no hardware modifications are usually required to run the system. This is true with the Paradox DBMS.

Off-The-Shelf products also arrive as complete packages which include tutorials, documentation and installation directions.

Training in the use of the Paradox DBMS is no different than with the DB3GEN product for programmers. Two weeks is the accepted norm after which programmers can begin to create applications. End users will require more training in use of the Paradox system than with the other two systems because of the ad hoc capabilities. The other systems do not require end users to do anything except follow the directions of the application created for them. Thus, two weeks of training, rather than one week, is required for end users.

The three costs for the Paradox system are the cost to initially purchase the system, updates to the system and training on system use. The cost of updating the system is not a yearly cost. On the average, Paradox-like systems distribute updates every two years. This figure was arrived at by averaging the update periodicity of the Paradox, FoxPro,

dBase and Informix DBMSs. So to arrive at a yearly figure, the cost to update all systems was divided in half to reflect the two year time cycle.

1. Financial Costs

Productivity measurements were made using Paradox version 3.0. It is no longer offered. The updated 3.5 version, which is presently on the market, retails for $795.00 per copy. The Local Area Network (LAN) version retails for $995.00. The update for version 3.0 retails for $135.95 per copy.

Paradox is compatible with all IBM Personal Computers (PCs) and requires no hardware alterations to run the program. The manufacturer claims to offer a site licensing agreement. This agreement does not allow any extra copies of the program to be made and only includes extra training benefits and telephone assistance. (Santana, 1991) This is not site licensing as desired by the Navy. (Hamblen, 1990, p. 6)

2. Performance Liabilities

Paradox does not produce Ada code. The NARDAC Norfolk successfully created an Ada/Paradox binder which allows a program written in Ada to call and use the Paradox program but still no Ada code is generated by the program. Paradox does provide all other Management Information System functions that DB3GEN and AdaSAGE presently offer. There is no reuse

capability of the Paradox code except for reuse of the entire application by different users.

3. Analysis

Training costs reflect two expenses. First, is the opportunity cost of the personnel being trained. Second, is cost of the instructor. Again using the rates given in the introduction, the training cost is determined as follows:

- 24 + one instructor
- One week of training
- 40 hours per person per week
- $32.76 per hour opportunity cost[2].

The product of these figures is $32,761.95. This expense will occur annually due to turnover of personnel.

Using the 33 Personal Computers (PCs) figure given in the introduction and a cost per copy of $795.00, the first year cost is the product $26,235.00. The annual cost of updating the system is 33 copies @ $195.00 and then divided over the two year periodicity. This comes to $3,217.50.

In summary, the costs of the Paradox DBMS are determined as in Figure 8.

---

[2]$32.76 equals the average of a Programmer/analyst's and a Junior programmer's hourly rate under NIF.

Training Costs

24 personnel * one week * 40 hours/week * 32.76/hour = $32,761

Software Costs

Purchase Cost

33 copies * $795.00/copy = $26,235

Update Cost

33 copies * $195.00/copy = $6,435 over a two year period

Total Costs

$32,761 + $26,235 = $58,995 for the first year

$3,217 for each year thereafter

Figure 8 - Paradox DBMS Costs

C. ADASAGE SYSTEM

The cost/benefit analysis of the AdaSAGE system is more complicated than with DB3GEN or Paradox. There are hardware modification and leasing considerations. Training is not only for the development tool but for a new language. The additional options such as a reusable code library must also be considered.

1. Costs

a. Training and personnel

The training costs for programming personnel vary according to the type of training. At the 1991 Ada Technical Workshop, the issue of training personnel was discussed and a consensus was arrived by comparing successful training initiatives carried out by various commands. As a minimum, each programmer requires basic training in the Ada language and in use of Ada development tools such as AdaSAGE. The basic language training would last two weeks and the tool instruction would last one week on the average. A bulletin board service has been established at NARDAC Norfolk that lists training facilities available and experiences learned by various commands. This is to help commands direct their perspective trainees toward the training facility that would most likely benefit their activity.

A typical training asset for the basic language education would be the Keesler Air Force Base Ada course. This training is at no cost to the requesting command besides the students' per diem and travel expenses. The school costs are provided for under a joint service training agreement. The problem is the limited number of seats available for training.

Typical tool instruction would come from a firm similar to EG & G who is under contract held by Naval Computer

and Telecommunications Command (NCTC) at a cost of $160,000 per year. Presently under this contract, only the personnel from Naval Computer and Telecommunication Station (NCTS) San Diego and NARDAC Norfolk are included because of a limited number of trainers. The training includes two weeks of instruction at each activity for approximately two dozen students. As more funding becomes available, more commands will be included.

Other NARDACs have sought Ada training from commercial sources. An example is NARDAC San Francisco. First, their senior programmer and two other personnel attended an introductory Ada language class. Upon completion of the course, they felt they were not prepared to create applications in Ada. To complete their training they contracted a local Ada consultant for $100 per hour for four hours per day four days a week for two weeks. According to a San Francisco representative at the 1991 Ada Technical Workshop, they gained more benefit in the first day with the contractor then they received from the entire course. This was attributed to the contractor being aimed towards creation of business applications and the introductory course being only for novice programmers.

In response to a NCTC questionnaire, seven of the eleven subordinate commands stated they required training of 452 personnel in the Ada language. Interpolating for the NCTSs that did not respond (Neither the largest nor the

smallest NCTSs failed to respond so interpolation would be valid.) puts the number of personnel to be trained at over 700. However those personnel are trained, the cost in terms of dollars spent on instruction and in terms of man days lost is significant. If all activities receive the level of training offered to NCTS San Diego, then over 7000 man-hours will be spent on training alone.

Training will also have to be conducted on the RAPID system. The Army's RAPID system has a one time installation fee that covers training of the users, training of the library custodian and installation of the system. This fee is $75,000.00.

Once the personnel are trained, the RAPID system requires the services of a full time library manager/custodian and an assistant. Using the rates for personnel under the NIF environment, the cost of the library manager/custodian and assistant will be $73,209.41 per year per person.

b. Cost of hardware

Each 286-based personal computer (PC) requires the addition of a two megabyte random-access-memory (RAM) expansion board in order to run the Alsys compiler. The cost of this expansion board is $597.44. (NAVCOMTELSTA San Diego catalog, 1991, p. 29)

The VAX computer and the associated accessories must also be leased for the Reusable Ada Products for Information

Systems Development (RAPID) system. The equipment necessary to run the RAPID system, costs $27,500.00 per year to lease. The installation cost for the hardware and the software is $75,000.00. (Rothrock, 1991)

c. Cost of software

There is no cost for the AdaSAGE System since it is a "shareware" program. But the amount of Government funds spent to develop the AdaSAGE system is in the millions. It was originally funded by the Department of Energy but the Army and Marine Corps have also contributed to the development costs. The number of program copies in use cannot be determined since it is legal to make as many copies of the program as users want. Therefore, it is impossible to determine an exact cost per copy of the program.

A cost that can be determined is the expense of the compiler that is required for each computer that runs the AdaSAGE program. Presently only the Alsys corporation has a compiler supports the full memory model on a Personal Computer (PC) that AdaSAGE requires and meets the specifications of American National Standards Institute (ANSI) / Military Standard 1815A. The cost of the Alsys compiler is $1815 and a one year maintenance contract that includes any upgrades that may be released during the year is $660. The United States Army has an open contract with Alsys under which compilers may be purchased at the reduced rate of $778.00 but

only for the 386 based PC system. One recommendation from the Navy's 1991 Ada Technical Workshop is to "untie" the AdaSAGE system from the Alsys compilers. This will require other contractors to spend money in the development and testing of a PC based full memory compiler. Other contractors will not make this investment until it is clear the DOD shows a serious commitment to Ada in the non-tactical PC environment. Until then, the Alsys organization has a virtual monopoly on the PC based Ada compilers.

The RAPID system also has software costs. The programs that run the VAX computer and operate the RAPID library are required. The software costs for RAPID is $28,250.00 per year to lease (Rothrock, 1991). Also required is a negotiable "maintenance fee" designed to cover their overhead costs of maintaining the system and quality control of individual procedures that are added to the library. This "maintenance fee" is supposed to be relative to the activity's usage rate of the Reusable Code Library (RCL). According to a customer service representative, the average cost of the "maintenance fee" is $125,000.00 per year. (Rothrock, 1991)

In summary, personnel, training, hardware and software costs for implementation of the AdaSAGE system with the RAPID library are as follows:

- "Maintenance Fee" - $125,000.00
- Hardware lease expense - $ 27,500.00

- Software lease expense - $ 28,250.00
- Installation expense (First year only) - $ 75,000.00
- Two megabyte expansion board per PC - $ 597.44
- One Alsys compiler per PC - $ 1,815.00
- One Alsys maintenance contract per PC - $ 660.00
- Training course - $ 80,000.00
- Librarian and assistant salaries - $146,418.82
- Installation fee (First year only) - $ 75,000.00.

2. Benefits

From the Ada productivity study mentioned in the AdaSAGE section, it is seen there is a 25 to 30 percent increase in productivity due to the use of the Ada language. There is also a 17 percent gain in productivity when a development tool is used. There is no indication of whether or not the productivity gains are additive. If they were, it would assist in reducing time to reach the break even point. For purposes of this analysis, they are considered additive for the following reasons:

- The study's reusage rate averaged only 18 percent while AdaSAGE has shown average reusage rates of 65 to 70 percent
- The learning curve decreases in productivity will not be as dramatic as in the study because the developers will being using a tool (AdaSAGE) and the learning process has already started at all of the activities.

These two issues point toward the AdaSAGE scenario being more favorable than in the study; so therefore the productivity gains of using the Ada language and an Ada development tool will be cumulative.

Assuming a 25 percent reduction in development costs, the average in the Ada productivity case study, and a 17 percent gain from the use of a development tool (AdaSAGE) the overall gain amounts to 42 percent.

3. Analysis

This analysis is for one activity only and the Navy wide costs are prorated for one activity. This includes cost of training and cost of hardware. While the actual costs will vary, this represents the best average estimate. Some activities will have more than five large (One million dollars or more) software development contracts others less. The larger activities will have more contracts but the development effort is spread out over more people. Smaller activities generally receive less contacts but have fewer analysts and programmers to rely upon. The point is no matter the size of an activity, the work load for each person is the same. Equal work loads mean equal time spent over coming the learning curve.

Activities will also vary on costs of contracts. Some will be larger than the one million dollar figure used in the analysis, others will be smaller. The relative losses and

gains in productivity are not dependent upon contract size. If an activity has larger contracts than the initial losses will be higher. But the eventual gains will be also larger. The opposite is true for organizations that receive small sized contracts. Thus activities which average small or large sized contracts will reach the break even point approximately the same time.

The discount factor, for taking into account the time value of money of 10 percent, is in accordance with Department of Defense Instruction 7041.3 and OMB Circular A-94. All other figures are as noted.

The first analysis, shown by Figure 9, assumes the productivity gains of using Ada and AdaSAGE are cumulative. According to the Reiffer study, the first five projects represented the learning curve process. The first three projects were developed with a 15 percent increase in cost. The next two projects were at the same cost as previous methods. The next four projects each had a ten percent reduction in cost until the average level of productivity improvement, 42 percent, was reached. The break even point is reached with the first project in the third year.

The second analysis, as shown by Figure 10, does not assume the productivity gains are cumulative. Therefore with the lower cost reduction, 25 vice 42 percent, the break even point takes longer to be reached. Here the break even point

or pay back period is delayed until after half way through the third year. The figures for both are as follows:

	YEAR 1	YEAR 2	YEAR 3	YEAR 4
Revenue				
Cost Savings	-0-	100,000	420,000	420,000
		200,000	420,000	420,000
		300,000	420,000	420,000
		400,000	420,000	420,000
		420,000	420,000	420,000
TOTAL	-0-	1,420,000	2,100,000	2,100,000
Avg. Disc. Factor	* 1	* .954	* .867	* .788
Present Value	-0-	1,354,680	1,820,700	1,654,800
Costs				
Learning Curve	450,000	-0-	-0-	-0-
RAPID (Hardware/ Software)	55,750	55,750	55,750	55,750
Personnel	146,418	146,418	146,418	146,418
Installation/ Maintenance	200,000	125,000	125,000	125,000
AdaSAGE Hardware	20,910	-0-	-0-	-0-
Software	86,625	23,100	23,100	23,100
Training	80,000	-0-	-0-	-0-
Total	1,039,703	350,268	350,268	350,268
Avg. Disc. Factor	* 1	*.954	*.867	*.788
Present Value	(1,039,703)	(334,155)	(303,682)	(276,011)
Net Present Value	(1,039,703)	1,020,525	1,517,018	1,378,789
Cumulative Total	(1,039,703)	(19,178)	1,497,840	2,876,629

Figure 9 -Cumulative Analysis

	YEAR 1	YEAR 2	YEAR 3	YEAR 4
REVENUE				
Cost Savings	-0-	100,000	250,000	250,000
		200,000	250,000	250,000
		250,000	250,000	250,000
		250,000	250,000	250,000
		250,000	250,000	250,000
TOTAL	-0-	1,050,000	1,250,000	1,250,000
Avg. Disc. Factor	*0	* .954	* .867	* .788
Present Value	-0-	1,001,700	1,083,750	985,000
COSTS				
Learning Curve	450,000	-0-	-0-	-0-
RAPID (Hardware/ Software)	55,750	55,750	55,750	55,750
Personnel	146,418	146,418	146,418	146,418
Installation/ Maintenance	200,000	125,000	125,000	125,000
AdaSAGE				
Hardware	20,910	-0-	-0-	-0-
Software	86,625	23,100	23,100	23,100
Training	80,000	-0-	-0-	-0-
TOTAL	(1,039,703)	350,268	350,268	350,268
Avg. Disc. Factor	* 1	*.954	*.867	*.788
Present Value	(1,039,703)	334,155	303,682	276,011
Net Present Value	(1,039,703)	667,545	780,068	708,989
Cumulative Total	(1,039,703)	(372,158)	407,910	1,116,899

Figure 10 - Non-Cumulative Analysis

In either scenario, the first year operates at a loss of $1,028,314. This is attributable to a decrease in productivity as the personnel goes through the learning curve process and the cost of the capital investment. In the second year, the learning curve has been overcome, the capital investments made and now the advantages of reusability and a development tool can be felt. In the third year, the break even point is reached and by the end of the fourth year the cost savings will surpass the initial investment in setting the system. See Figure 11 for the summary.

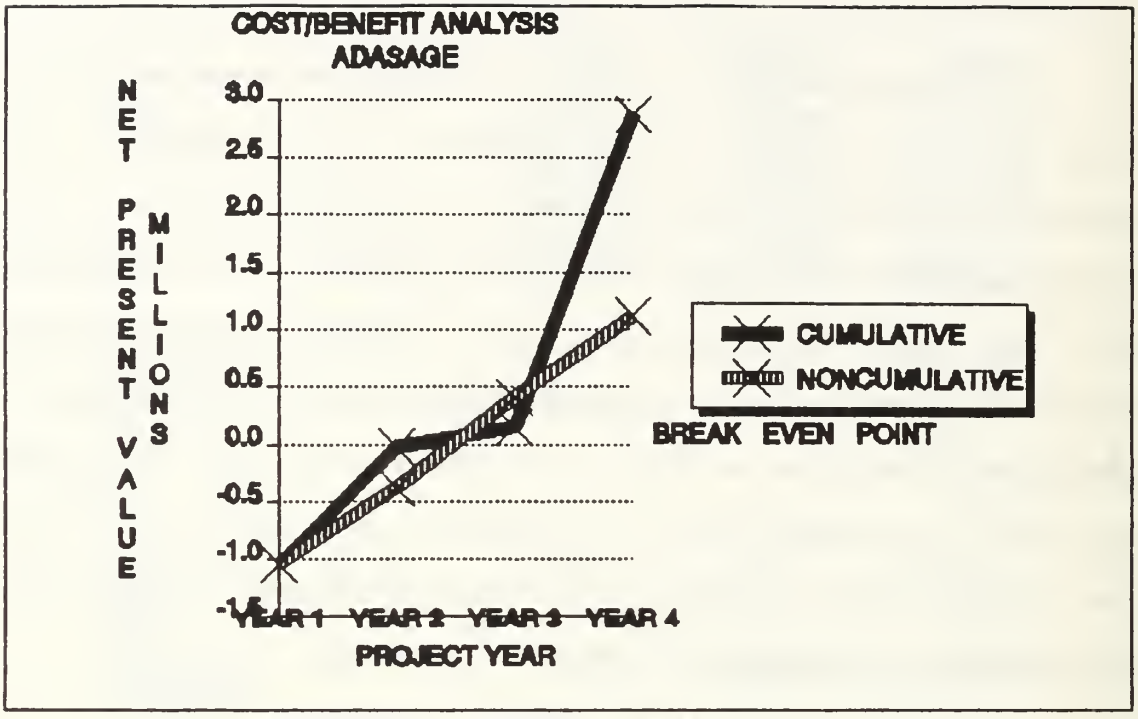

Figure 11 - AdaSAGE Cost Benefit Analysis

VII. RESULTS, CONCLUSIONS AND RECOMMENDATIONS

A. OVERALL RESULTS

1. DB3GEN Results

DB3GEN cannot compete against commercial products when it is funded via the Naval Industrial Fund (NIF) concept. End users do not have the Information Resources (IR) funds to pay for product enhancements. Commands senior to Naval Computer and Telecommunications Station (NCTS) San Diego are prohibited by directive and regulation from funding the product. Even if they were not, it is more economical to purchase Off-The-Shelf. The cost of maintaining a full time development and support staff is higher than purchasing 334 copies of Paradox 3.5 the first year. After the first year, the cost of Paradox decreases due to only having to purchase update packages, if they are released during the year, but the cost of DB3GEN does not decrease.

In addition to the added cost of developing and maintaining DB3GEN, another negative factor is the decreased functionality it offers programmers and end users in comparison to Paradox 3.5 like products. DB3GEN does not offer a Local Area Network (LAN) version, complete end user ad hoc capabilities, graphics, pop-up windows and secret level security functionality. The existing documentation needs

improvement in readability and completeness. In order for the user to create data validation statements and combination sorting instructions, dBase III+ programming statements must be coded. The DB3GEN system does not offer a library of reusable code. It does not generate Ada code and is presently not supported by a binder for use in an Ada program.

2. Paradox Database Management System Results

The Paradox database system at the Norfolk Naval Regional Data Automation Center (NARDAC) has been an unqualified success for both end users and programmers. For end users, it has met all their requirements and given them ad hoc capabilities. The ad hoc capabilities allow for faster turn-around times when creating new reports and graphs. Most importantly, they allow for changes in end user requirements. When a need changes or a new requirement arises, end users can make changes without assistance from analysts and programmers.

Paradox has also been a success for programmers. They took a chance by adopting the Paradox system. They could have remained secure in their positions using traditional C, COBOL and Clipper languages. Yet, in just two months, two programmers were able to do the work of six in one third the time. The product they delivered had fewer errors and the errors they had were easier to find and correct than before.

3. AdaSAGE System Results

The literature which accompanies the AdaSAGE demonstration disk goes to great lengths to tell users AdaSAGE is not a database management system. The AdaSAGE developers admit that the primary use of AdaSAGE to date has been that of a database application generator but it actually is an application development environment. The difference may not be readily apparent but it is what holds great promise for the future use of this system.

As a database application generator, it does not yet provide all of the benefits of a fourth generation language based database system. It does include a menu-driven database development process complete with an automatic code generator but it does not yet include ad hoc capabilities for the end user as with the Paradox system. These capabilities allow end users to be independent of system analysts and programmers. This factor cannot be understated as the proponents of both the AdaSAGE system and the Clipper system at NARDAC Norfolk claim. Both of these system proponents state 95 percent of end users do not want to have ad hoc capabilities. All end users of the THAIS system and the Marine Corps logistics systems at the Albany, Georgia Marine Corps Depot stated they would want ad hoc report, form, graphics and query capabilities if they were offered. The AdaSAGE software development methodology (AdaSAGE-SDM) relies upon rapid prototyping to identify and include all end user requirements

in the application as it is developed. But the system requirements will change over time no matter how many iterations of prototyping are conducted. If end users have the capabilities to change with the changing needs, then the system will continue to provide the needed data without any interruption. If end users have to turn over the system to developers and programmers every time their requirements change then development and turn-around time will be wasted time for end users. If they had ad hoc capability then they could change the system by themselves.

AdaSAGE, with the use of RAPID, has a greater growth potential than commercial database systems. Each newly coded module adds potential to the AdaSAGE system for reusability. Commercial databases do not offer the modularity required to achieve the same level of reusability.

Another advantage AdaSAGE offers that other DBMSs do not is the ability to tie into embedded systems. True integration of processes that combine embedded and transactional systems cannot be accomplished with the commercially available fourth generation language based database systems alone. They must be combined with a third generation language that means having either a separate team of programmers or contracting out the non-database section(s).

A major area of concern in the proposed switch to the AdaSAGE system is the decreased ad hoc ability of the end user. Typical ad hoc capabilities that an end user would have

on a fourth-generation-language-based database system that includes automatic code generation features are many. They include report generation, form generation, graph generation, the addition or deletion of data fields, the establishment of different relationships between data fields, the sorting of the data on various attributes and the customization of the screens and menus that lead the user through the database application. The AdaSAGE environment does support all of these features except graphics but end users cannot make these changes. The changes must be submitted to the analysts and programmers.

The AdaSAGE developers have tried to decrease the changes end users might otherwise need by including rapid prototyping in the development phase. The initial changes will decrease but according to end users interviewed, after about one month they found new requirements for the system. But since end users did not have ad hoc capabilities, analysts and programmers were required to make the changes. Had the system contained ad hoc capabilities end users could of made the changes themselves.

This problem has been addressed by the creators of AdaSAGE by the inclusion of the ad hoc type packages in the library. The QBE is an example of a package designed to meet this need. This package is designed to allow the user to define relationships based on the join operation. The name of this package might lead the user to believe it was the same

Query By Example as created by the Borlan Corporation but this is not true. But according to experienced AdaSAGE programmers, end users without depth training would not be able to use the QBE package without programmer assistance. (Sugar, 1990)

The use of the AdaSAGE system by developers gives them an advantage over other systems that do not use reusable code. Both development time and maintenance time is decreased by use of reusable code. The use (or reuse) of newly written packages is a very quick process. In the best case, all that is required is the inclusion of a "procedure call" in the application. This comes after finding a procedure to satisfy the requirements of the application being created. The most likely scenario is a previously written application must be modified to meet new requirements. Once it has been adapted a procedure call is written into the application and then the new application can be included in the Navy's library for future reuse. Development systems that do not reuse code must start from scratch each time an application is created or modified.

The true value of the AdaSAGE system lies in three areas. The first is for the development of embedded systems. This was the original reason for the creation of the Ada language. The language is strongly structured and fully supports object-orientated programming. This aids in creation of documentation for a system and in life cycle maintenance

costs because object orientated programs are modular in design and therefore easier to maintain. That this language also can be used to create transactional systems is a tribute to it's flexibility but its original purpose is for embedded code.

The second advantage of the AdaSAGE system is it produces code in the Ada language. This is an advantage that cannot be taken lightly. Since 1983 Ada has faced stiff opposition in its acceptance among non-tactical programmers. Resistance to any change has been well documented by psychology studies (Herbert, 1976, pp. 342-344, 430-433). The switch to the Ada language is not different. Especially when programmers are rated, along with other attributes, by their familiarity and years of experience in use of a language. But by commitment of senior Navy personnel and education of programmers on the benefits of Ada, it has gained in use and popularity.

The third advantage of AdaSAGE is it allows reuse of previously coded modules. Establishment of reusable libraries is a matter of high concern for Department of Defense (DOD) activities. The United States Army, Air Force and Marine Corps have already begun formulating their libraries. The Army's RAPID system is the most advanced. It is also the model the Navy should use in designing its own library.

B. RECOMMENDATIONS

1. Productivity

The one constant misunderstanding between all end users and all programmers interviewed, with the exception of the two Paradox programmers and one Paradox systems analyst, is the role of ad hoc capabilities by the end users. Systems analysts for DB3GEN at NARDAC San Diego and Clipper at NARDAC Norfolk stated 95 percent of all end users do not want ad hoc capabilities. The AdaSAGE analysts agreed with these analysts that if the needs of end users are properly identified during the system development phase, then the need for end user ad hoc capability should not exist.

Contrary to this idea is the desire to have ad hoc capability by all end users who were interviewed. One end user, who had recently received ad hoc capability for the first time, went as far as to say he could not imagine going back to the way he had operated before (without ad hoc capability). In the questionnaires given to both programmers and end users ad hoc capability had the third greatest difference in terms of rated level of importance of all the characteristics evaluated.

End users are becoming better educated in regard to the use of computer systems. The users of the information which Management Information Systems (MIS) supply are also becoming better educated and are demanding more from their

systems. Programmers must realize this and stop delivering applications that require end users to operate in only one manner.

2. Development Tool Procurement

This thesis examined programs from the three different sources of non-tactical software: In-House developed, Off-The-Shelf, and a combination of outside contractor and DOD asset developed. Each method of acquisition offers advantages over the others.

a. In-House developed

In a present day analysis, the Off-The-Shelf product offers the best value for the United States Navy. This product, Paradox, is by no means unique in what it offers. But what it and other similar products do offer, out weigh the other two options.

The In-House developed program costs too much compared to the other two options. A full time staff must be applied to maintain and improve the program. Even when this staff is maintained at an unrealistically low level of a middle level analyst and a middle level programmer, it costs more than the Off-The-Shelf option. The question who would fund the staffing under the Naval Industrial Funding (NIF) concept is a matter to be determined.

The staff would be constantly under pressure to perform as well as the civilian sector. As competitive

products offer more functions and capabilities the In-House developers would be tasked to provide the same. Commercially funded systems either adapt and improve or lose their market position. The Navy's system under the In-House or combined options will adapt or modify only when the need is identified and funded. As the funds available for the Information Resources (IR) community decrease, the ability to compete with the commercial developers will decrease.

This is not a problem as long as the needs of the Navy coincide with the marketplace needs. But when the Navy has a requirement the marketplace does not have, it falls back on the Navy to pay for the modification or development. This problem is lessened when the four services combine their resources and jointly fund the development of a tool such as AdaSAGE. But then the problem of coordinating what changes to make arise.

An advantage this alternative has is immediate responsiveness to the needs of the Navy. Once a need is identified, all that is required to make the change or changes is for the Navy's analysts and programmers to design and develop the system or application. No bidding or convincing an outside contractor it is in their interests to develop or adapt a product to meet the needs of the Navy is required.

b. Contractor developed

Contractor supplied assets have many advantages over both the Off-The-Shelf and In-House options. They have the responsiveness of the In-House method since both the military and contractor assets answer to DOD guidance. They allow expertise to be "imported" from contractors and combined with the functional experience of military personnel. It allows flexibility to switch contractors if the performance of the present organization is not up to required levels. In short, it offers the advantages of the Off-The-Shelf option with increased DOD input and influence.

The disadvantage of the combination option is the product presently being offered by use of this alternative. This product is the AdaSAGE development environment. The primary disadvantage of the AdaSAGE program is not the Ada language, the personnel developing and maintaining the product, or the identification of shortcomings in the system. These actually are all advantages. The disadvantage this system has is two fold and the two problems are interrelated.

First, the AdaSAGE program is presently in a "tail chase" scenario where it is constantly trying to develop the functionality already being offered by the Off-The-Shelf market. The development of the Local Area Network (LAN) system called Multi-SAGE and the ad hoc query system called QBE are examples of this reaction by the combination assets to

provide functionality already offered by commercial products. Until the AdaSAGE system equals the functionality of the marketplace products it cannot lead them.

The second, but related disadvantage is funding. Making improvements and modifications to AdaSAGE requires funding. Under the Navy Industrial Funding (NIF) concept, a client must pay for all work conducted by analysts, developers and programmers. No user client has the Information Resources (IR) funding to sponsor an update to the whole AdaSAGE system and the NCTSs are forbidden to initiate changes under the NIF concept. User clients only have limited funding for creation of specific applications for their own activity. Changes for the entire AdaSAGE system must come from a service-wide, Department of Defense-wide or even United States' government-wide activity.

The problem is client activities and NCTSs are the users of AdaSAGE and therefore the people who identify needed changes. But these activities do not have the funding to make the changes occur. The parent commands, who do not directly use the tool, do have the funding capabilities. Communication of needed changes then becomes a critical element of having a successful tool. The subordinate commands must inform their superiors when a need arises AdaSAGE can but does not satisfy. It is then the responsibility of the parent command (NCTC for example) to have foresight to allocate limited funding toward only those changes needed and will benefit the most users.

So far this is not a problem. The Ada Technical Workshop sponsored by the Navy Computer and Telecommunications Command (NCTC) ironed out the initial Navy policy on updates to the AdaSAGE system. This was determined after receiving inputs from the three other services and all NCTSs. One of the recommendations was the formation of a configuration board that would monitor and plan changes to the AdaSAGE system to ensure the needs of the Navy are met.

c. Off-The-Shelf developed

The Paradox 3.5 system offers the most value of the three options. It is the only option that offers all important characteristics as identified by programmers and end users. It has a proven record of improving productivity. It has the least expense life cycle costs over the two years the costs can be accurately estimated. If a system was to be purchased for the development of a system over just the next two years, the Paradox 3.5 or a comparable Database Management System (DBMS) which has all the functionality of the Paradox System should be selected.

The Paradox system has two primary advantages over the competing systems. First, it offers ad hoc capabilities to end users. This is the basis for the productivity improvements. The requirement for analysts and programmers to make report, form, graph or screen changes to a delivered system is removed. This lowers the maintenance costs and work

load. These capabilities also shorten the development time required to identify end user requirements for outputs. If a requirement changes, the end user makes the change.

The second advantage the Paradox system has is services the Navy does not have to fund. Foremost among these services are updates and changes to the system. As new functionality is added or as defects are identified and corrected, the Navy does not have to contribute to the funds. This is at the sacrifice of controlling the direction of the updates and changes. This sacrifice is lessened because the other two products are trying to update to the present level of the Paradox system. Another service the Navy does not have to fund is the dial-in phone service for questions and answers.

Disadvantages of the Paradox system come from the comparison of it versus the AdaSAGE system. It does not produce Ada code and therefore does not meet the guidelines of the Congress and the DOD. It does not offer a reusable library of procedures and the possibility of creating one are remote due to limited nature of its applications: databases only. It is not compatible with embedded code. This further limits it's uses in integrating systems.

All of these disadvantages limit the future growth of the Paradox system with regard to it becoming a DOD wide tool or application generator and this is where the DOD should address its efforts. With all of the advantages Paradox has

over AdaSAGE, its lack of growth potential makes it a poor choice for investment purposes in the long term. Here, the long term is meant by any time over three years. In one year the learning curve will be overcome and then the benefits of reusability can be realized. In two years the ad hoc capabilities, now missing, can be developed. In the third year all the advantages a Paradox like system has over AdaSAGE will be gone and AdaSAGE will be enjoying the reusability benefits others do not have. Paradox and similar systems will continue to grow during this two year time frame but they can act as guides and test beds for showing the way towards future AdaSAGE improvements. Meanwhile, the library of reusable code will grow and save the DOD millions in development costs. Today the Paradox system has the advantage but tomorrow the AdaSAGE system will have more value.

3. Purchasing Strategies

Site licensing is paying one fee for as many copies of an application as the site requires. Site licensing is also a high priority issue as evidenced by Vice Admiral Tuttle's interview in the July issue of CHIPS magazine distributed by NARDAC Norfolk.

> "I want site licensing for the Navy. And I'm going to discourage using the software manufacturers who don't come aboard." (Hamblen, 1990, p. 6)

The normal method for paying for the programs is by the copy. At a software development facility that conducts its programming on PCs it rapidly becomes expensive when each computer requires a copy of a certain program in order for each programmer to create an application. Local Area Network versions do not help since the development program has only one set of the files necessary to create applications.

As an example, one Navy activity, which has 100 copies of the Clipper database management program, is required to purchase 100 copies of the updated program when it upgrades to the 5.0 version of Clipper. The cost of one copy was $207.00. The personnel at the command hoped to get a volume discount but had not asked the Clipper corporation even if they had a site licensing policy.

This was brought out as an example and not to single out a particular command. In fact, from a questionnaire in regard to the number and type of database programs at all of the NARDACs and NAVDAFs (Naval Data Automation Facilities), only two activities (NARDAC San Diego and NAVDAF Orlando) had a site licensing agreement on any database program.

The entire Navy needs to speak as one voice to the software manufacturers in order to put authority behind its demand for site licensing. The individual commands cannot bring enough pressure to bear to force the issue. The best scenario would be a joint contract of DOD licensing of products. The worst this can do is bring attention of senior

management to the overall expense of dealing with the purchase of programs individually on a service wide basis.

C. CONCLUSIONS

These conclusions are lessons learned from working with and most importantly, observing non-tactical software developers in DOD. They are guidance for all managers of non-tactical software development assets.

- Ad hoc capabilities improve end user and programmer productivity. All applications delivered to Naval end users should have ad hoc capabilities.
- Use and invest in the AdaSAGE system. It is not the best system presently available and it is actually a step backwards in technology. But it uses the Ada language and has the most promise for future. After the third year, productivity increases pay for the system.
- The number one priority for improving AdaSAGE should be the development of complete ad hoc capabilities end users can easily use in delivered applications.
- If any tool is developed which combines both Ada code generation and ad hoc end user capabilities, buy it.
- Promote competition in the area of Ada microcomputer compilers. A sole source supplier of compilers stifles creativity and does not lead to a fair market price.
- Adopt the United States Army's RAPID system as the Department of Defense wide Ada procedure repository. A jointly funded and manned library goes a long way towards showing a strong commitment to the Ada language in the non-tactical arena.
- Strive for site licensing at the Department of Defense level to obtain volume discounts and better manage Information Resource assets. The buying power of the entire Department of Defense will make contractors see the need for site licensing.

D. FOLLOW-ON STUDY AREAS

This thesis has found as many unanswered questions as it has solved. The following is a list of areas which need to be addressed:

- A case study of one or more applications developed using AdaSAGE or any other Ada-based non-tactical software development tool. Are the productivity gains the same as forecast?
- A case study involving productivity at DOD activities that have instituted function point analysis. Is function point analysis as accurate with Ada?
- A case study of planning and procurement procedures within the non-tactical software community. Specifically, is effective planning being conducted to support required policy issues?

LIST OF REFERENCES

1. Arthur, L. J., *Measuring Programmer Productivity and Software Quality*, John Wiley & Sons, 1985.

2. Awad, E. M., *Management Information Systems*, The Bejamin/Cummings Publishing Company, 1988.

3. Boehm, B. W., *Software Lifecycle Factors*, Van Nostrand Reinhold Company, 1984.

4. Boehm, B. W., *Improving Software Productivity*, IEEE Computer, 1 September 1987.

5. Chorafas, D. N., *Fourth and Fifth Generation Programming Languages, Vol. II*, McGraw-Hill Book Company, 1986.

6. Conte, S. D., Dunsmore H. E., and Shen V. Y., *Software Engineering Metrics and Models*, The Bejamin/Cummings Publishing Company, 1986.

7. Dreger, J. B., *Function Point Analysis*, Prentice Hall, 1989.

8. Eisner, H., *Computer-Aided Systems Engineering*, Prentice Hall, 1988.

9. Fairley, R. E., *Software Engineering Concepts*, McGraw-Hill Book Company, 1985.

10. Hamblen, D., "VADM Tuttle: Sailing Into the 21st Century", *Chips*, Volume X Issue 3, July 1990.

11. Herbert, T., *Dimensions of Organizational Behavior*, Macmillan Publishing Company, 1976.

12. Johnson, P. I., *The Ada Primer*, McGraw-Hill Book Company, 1985.

13. Jones, C., "Using Function Points to Evaluate CASE Tools, Methodologies, Staff Experience, and Languages", *CASE Trends*, January/February 1991.

14. Levine, T., "Reusable Software Components", *Ada Letters*, Volume X Number 5, May/June 1990.

15. Martin, E. W., *Strategy for a DOD Software Initiative*, IEEE Computer, 1983.

16. Martin, J., *Application Development Without Programmers*, Prentice-Hall, 1982.

17. Martin, J., *Fourth-Generation Languages: Volume I, Principles*, Prentice-Hall, 1985.

18. Nelson, R., *End User Computing: Concepts, Issues and Applications*, John Wiley & Sons, 1989.

19. Panko, R. R., *End User Computing: Management, Applications, and Technology*, John Wiley & Sons, 1988.

20. Piper, J., and Barner W. L., *RAPID Systems Operator Guide*, U. S. Army Information Systems Software Development Center-Washington, 1990.

21. Reifer, D., *Softcost-Ada: User Experiences and Lessons Learned at the Age of Three*, Association for Computer Machinery, Inc., 1990.

22. Robb, D. W., "Product Preference Survey: Paradox DBMS Has an Edge Over Ubiquitous Ashton-Tate", *Government Computer News*, 28 May 1990.

23. Sackman, H. and others, *Exploratory Experimental Studies Comparing On-Line and Off-Line Programming Performance*, Communications of the ACM, 1968.

24. Schatz, W., "The Pentagon's Botched Mission", *Datamation*, Volume 35, 1 September 1989.

25. Shelly, G. B. and Cashman T. J., *Computer Fundamentals for an Information Age*, Anaheim Publishing Company, 1984.

26. Simpson, A., *Understanding dBase IV*, Sybex Inc., 1989.

27. Stewart, H., "AdaSAGE", *NCTC Ada Technical Workshop*, 1991.

28. Taylor, M., *AdaSAGE- An Applications Development System for Ada*, Idaho National Engineering Laboratory unpublished pamphlet, 1990.

29. Wallace, R. H., *Practioner's Guide to Ada*, McGraw-Hill, 1986.

30. Yourdon, E., *Managing the Life Cycle*, Yourdon Press, 1988.

31. Telephone conversation between Sergeant Schugar, USMC, Marine Logistics Command, Albany Ga., and the author 15 November 1990.

32. Telephone conversation between Mr. Jack Rothrock, Softtech: Prime contractor for U. S. Army's RAPID system, and the author, 12 February 1991.

33. Telephone conversation between Mr. Frank Santana, Borland Corporation Government Sales Department Head, and the author, 15 February 1991.

34. NAVCOMTELSTA San Diego, *Value Added Computer Store Price List*, 1 February 1991.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center — 2
   Cameron Station
   Alexandria, Virginia 22304-6145

2. LCDR Robert Knight, USN — 1
   Code AS/KT
   Naval Postgraduate School
   Monterey, California 93943-5002

3. Prof. Terek Abdel-Hamid — 1
   Code AS/AH
   Naval Postgraduate School
   Monterey, California 93943-5002

4. Prof. Tung Xuan Bui — 1
   Code AS/BD
   Naval Postgraduate School
   Monterey, California 93943-5002

5. Prof. William James Haga — 1
   Code AS/HG
   Naval Postgraduate School
   Monterey, California 93943-5002

6. Library, Code 52 — 2
   Naval Postgraduate School
   Monterey, California 93943-5002

Thesis
L6435 Lindenbaum
c.1 Enhanced productivity
tools.